

## PROCESSING NUMERIC ADDRESSES IN A NETWORK ROUTER

### TECHNICAL FIELD

The invention relates to computer networks and, more particularly, to address processing in computer networks.

### BACKGROUND

A computer network is a collection of interconnected computing devices that can exchange data and share resources. In a packet-based network, such as the Internet, the computing devices communicate data by dividing the data into small blocks called packets, which are individually routed across the network from a source device to a destination device. The destination device extracts the data from the packets and assembles the data into its original form. Dividing the data into packets enables the source device to resend only those individual packets that may be lost during transmission.

Certain devices, referred to as routers, maintain tables of routing information that describe routes through the network. A “route” can generally be defined as a path between two locations on the network. Upon receiving an incoming data packet, the router examines destination information within the packet to identify the destination for the packet. Based on the destination, the router forwards the packet in accordance with the routing table.

A router is associated with a numeric address, for example, an Internet Protocol (IP) address. This numeric address identifies the router and can be output by any of a number of system modules, such as Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF) protocol modules. Generating numeric addresses in command output is useful and simple, and avoids problems inherent in name resolution. In many cases, however, the end user would be better served by displaying the symbolic names associated with the numeric addresses rather than the numeric addresses themselves.

In some routers, name resolution is handled by the module that generates the numeric address. The module typically queries a name server, *e.g.*, a domain name server (DNS), which resolves the numeric address to a symbolic name. While this approach will usually produce a name, the process is time-consuming because the name server is typically not located in the same place as the router. The delay introduced by querying the name server

can lead to packet blocking problems while the address is being resolved. Further, other activities, such as updating the router table, are delayed until the name is resolved.

Additional problems may result if the name server is unavailable or not present.

Other routers use proxies to report the address information, rather than the modules themselves. While this approach may relieve some of the burden on the modules, it involves a more complex architecture.

## SUMMARY

In general, the invention facilitates processing of numeric addresses by directing the user interface, rather than the system modules, to handle the task. Accordingly, a user-specific module processes the addresses, thereby avoiding packet blocking problems while avoiding the need for more complex architectures. Relieving the system modules from this task allows the system modules to process other tasks, improving overall system efficiency. In addition, the need to build code for handling addresses into the system modules is eliminated. Instead, only the user-specific module includes the code.

Certain embodiments of the invention may involve encoding numeric address information using an markup language, such as XML. XML is one example of an markup language in the class encompassed by the Standard Generalized Markup Language (SGML) specification, and will be described herein for purposes of illustration.

In one embodiment, the invention is directed to a method in which a router receives output in a format describing a type of the output. A server selected as a function of the type of the output is queried, and the router provides a response from the server to a user.

Another embodiment is directed to a method for processing an address in which a numeric address is received in a self-describing format. A name server is queried to resolve the numeric address to a symbolic name. The symbolic name is provided to a user.

Still another embodiment is directed to a method for processing an address in which a command is received in a user interface module. A system module is invoked to process the command and produces an XML-tagged IP address. A domain name server is queried to resolve the IP address to a symbolic name, which is provided to a user.

Other embodiments are directed to processor-readable media, apparatuses, and systems for performing these methods. The details of one or more embodiments of the

invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

5

## BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram illustrating an example router configured consistent with the principles of the invention.

FIG. 2 is a block diagram illustrating an example routing engine consistent with the principles of the invention.

10

FIG. 3 is a block diagram illustrating a network router incorporating a CLI module that supports address resolution according to the principles of the invention.

FIG. 4 is a flow diagram illustrating a mode of operation of performing address resolution consistent with the principles of the invention.

15

## DETAILED DESCRIPTION

A router facilitates processing numeric addresses by directing a user interface, rather than system modules, to handle processing. Addresses are processed by a user-specific module, avoiding packet blocking problems while maintaining a relatively simple software architecture. Relieving the system modules from this task allows the system modules to process other tasks, improving overall system efficiency. In addition, the need to build address handling code into the system modules is eliminated. Instead, only the user-specific module includes address handling code. Code development is thus simplified.

20

In this discussion, various embodiments of the invention are described in the context of resolving numeric addresses to symbolic names. The principles of the invention, however, are generally applicable to any application involving using a remote server to obtain a numeric address. Some such applications involve tasks other than name resolution. For example, the techniques described herein can be used to identify an owner associated with an address or to determine router policies associated with an address.

25

FIG. 1 is a block diagram illustrating an example network router 10 appropriate for resolving numeric addresses into symbolic names in accordance with the principles of the invention. Network router 10 receives and forwards data packets across a network. As

30

shown in FIG. 1, router 10 includes a control unit 12 with a packet routing engine 14 and a packet forwarding engine 16. Router 10 also includes one or more interface cards (IFCs) 18 for receiving and sending data packets via network links 20 and 22, respectively. Control unit 12 routes inbound packets received from inbound link 20 to the appropriate outbound link 22. Control unit 12 routes packets according to routing information stored in routing table 21.

Routing engine 14 maintains and updates the routing information within routing table 21. Forwarding engine 16 analyzes the contents of routing table 21 prior to receiving packets and pre-selects routes to be used when forwarding packets. Forwarding engine 16 then stores the selected routes in forwarding table 23. Upon receiving an inbound packet, forwarding engine 16 examines information within the packet to identify the destination of the packet. Based on the destination, forwarding engine 16 selects an available route and forwards the packet to one of the IFCs 18. IFCs 18 may be configured according to one of several different network protocols.

Packet routing engine 14 contains system modules, at least some of which generate numeric addresses as output. These system modules may include, for example, BGP and/or OSPF protocol modules. Other system modules include, but are not limited to, firewall filters. Consistent with the principles of the invention, the data output by these system modules is self-describing. At least some of the system modules generate numeric addresses that are tagged using, *e.g.*, XML tags, to identify them as addresses. For example, the numeric address 10.0.0.1 might be tagged using the following format:

```
<address> 10.0.0.1 </address>
```

Other modules can then resolve the tagged numeric addresses into symbolic names, for example, by invoking a command line interface (CLI) that queries a domain name server (DNS) to determine the name associated with a numeric address. Accordingly, the task of resolving numeric addresses to symbolic names is shifted from the system modules to the command line interface, thereby freeing the system modules to handle other requests.

Router 4 may include, or be used in conjunction with, some form of computer-readable media. By way of example, and not limitation, computer readable media may comprise computer storage media and/or communication media. Computer storage media includes volatile and nonvolatile, removable and nonremovable media implemented in any

method or technology for storage of information such as processor-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, random access memory (RAM), read-only memory (ROM), EEPROM, flash memory, CD-ROM, digital versatile discs (DVD) or other optical storage, magnetic cassettes, 5 magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to store the desired information and that can be accessed by router 4.

Communication media typically embodies processor readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport medium and includes any information delivery media. The term "modulated data 10 signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media, such as a wired network or direct-wired connection, and wireless media, such as acoustic, RF, infrared, and other wireless media. Computer readable media may also include combinations of any of the media described 15 above.

FIG. 2 is a block diagram illustrating an example router control unit 12 incorporating a command line interface consistent with the principles of the invention. In the example of FIG. 2, control unit 12 includes packet routing engine 14 and packet forwarding engine 16. Within control unit 12, routing engine 14 provides a management interface that interacts with 20 a number of software modules running within an operating environment provided by operating system 24. Operating system 24 provides a multi-tasking operating system for execution of a number of concurrent processes. An example of such an operating system is FreeBSD, which is an advanced UNIX operating system that is compatible with a number of programmable processors (not shown), including processors commercially available from 25 Intel Corporation.

As examples, routing engine 14 may include a chassis module 26, a device configuration module 28, and a routing protocol module 30 running as processes within the operating environment of operating system 24. A management server module 32 provides a user interface for interaction with system modules 26, 28, 30. Chassis module 24 defines an 30 inventory and status of components installed in the chassis of network router 10, including IFCs 18. Device configuration module 28 defines and controls the physical configuration of

ROUTER CONTROL UNIT

network router 10. Routing protocol module 30 administers protocols supported by network router 10. For example, routing protocol module 30 may implement protocols for exchanging route information with other routing devices and for updating routing table 21 (shown in FIG. 1).

5 Management server module 32 communicates with one or more client interface modules running on routing engine 14. In the example of FIG. 2, management server module 32 communicates with a command line interface (CLI) module 34. Command line interface module 34 serves as a daemon process that listens for requests from network router clients. The clients may take the form of human users such as system administrators or automated 10 script applications. Initially, command line interface module 34 listens for CLI commands, and passes them to management server module 32 for handling. The command line interface presented by control unit 12 can be dynamically replaced with an XML-based API upon receipt of a particular CLI command from a client. More specifically, management server module 32 redirects incoming commands from CLI 34 and services them based on the XML-based API.

15 XML is one example of an extensible markup language in the class encompassed by the Standard Generalized Markup Language (SGML) specification, and will be described herein for purposes of illustration. The official XML specification is governed by the World Wide Web Consortium and is available on the World Wide Web at <http://www.w3.org/TR/REC-xml>. The structure of the XML tags communicated via the 20 XML API may be defined using Data Type Definition (DTD) files, XML Schema Language files, or other similar devices for XML tag definition. As an example, the XML tags may conform to the evolving JUNOScript™ API developed by Juniper Networks, Inc. of Sunnyvale, California. The JUNOScript™ API is described in JUNOScript™ API Guide and 25 Reference, Version 4.4, available from Juniper Networks, Inc., the entire content of which is incorporated herein by reference.

FIG. 3 is a block diagram illustrating a network router incorporating a CLI module that supports resolution of numeric addresses into symbolic names, consistent with the principles of the invention. In the example of FIG. 3, management server module 32 accesses one or more system modules 36 running on routing engine 14, as well as other router 30 resources, such as database 38, to serve client requests. System modules 36 may include a

ROUTING ENGINE 14

10

15

20

25

30

35

40

45

50

55

60

65

70

75

80

85

90

95

100

105

110

115

120

125

130

135

140

145

150

155

160

165

170

175

180

185

190

195

200

205

210

215

220

225

230

235

240

245

250

255

260

265

270

275

280

285

290

295

300

305

310

315

320

325

330

335

340

345

350

355

360

365

370

375

380

385

390

395

400

405

410

415

420

425

430

435

440

445

450

455

460

465

470

475

480

485

490

495

500

505

510

515

520

525

530

535

540

545

550

555

560

565

570

575

580

585

590

595

600

605

610

615

620

625

630

635

640

645

650

655

660

665

670

675

680

690

695

700

705

710

715

720

725

730

735

740

745

750

755

760

765

770

775

780

785

790

795

800

805

810

815

820

825

830

835

840

845

850

855

860

865

870

875

880

885

890

895

900

905

910

915

920

925

930

935

940

945

950

955

960

965

970

975

980

985

990

995

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

variety of software modules such as chassis module 26, device configuration module 28, and routing protocol module 30 described with reference to FIG. 2. System modules 36 may also include a BGP protocol module, an OSPF protocol module, and/or a firewall filter module.

Database 38 may store information associated with system modules 36, such as router configuration information and operational information. A management interface schema 40 maps extensible markup language tags received by management server module 32 to information associated with system modules 36, including the information in database 38 and information that may be obtained directly from system modules 36. Schema 40 may take the form of a data definition language (DDL) file, and can be stored on a hard disk or other data storage medium.

Management server module 32 presents an XML API 42. A variety of network router clients, such as a CLI client 44, may establish independent communication sessions with management server module 32 using CLI module 34. CLI client 44 may take the form of a remote computer operated by a human user who enters CLI commands encoded with XML tags that conform to the API. In this case, the client application running on CLI client 44 may render the XML output received from management server module 32 as command line output, *e.g.*, in a textual form.

One or more of the system modules 36 provide output to CLI module 34. This output includes numeric address information, such as IP addresses. For example, in response to the command

```
show BGP neighbors -name
```

a BGP protocol module may return a list of network peers identified by their respective IP addresses. A user, such as a human user operating a remote computer of CLI client 44, would often find it more useful to receive a list of network peers identified by symbolic names, rather than numeric IP addresses. Accordingly, as described below in connection with FIG. 4, CLI module 34 queries a name server 46, such as a domain name server (DNS), to resolve the numeric address information to a symbolic name. To facilitate name resolution, system module 36 outputs the numeric addresses in a self-describing format using, for example, XML tags.

Using CLI module 34 to perform the address resolution relieves the BGP protocol module from this task and the delays associated with querying name server 46. As a result,

the BGP protocol module is available to process other requests. More generally, CLI module 34 may handle name resolution for all system modules 36, allowing them to perform other tasks.

FIG. 4 is a flow diagram illustrating a mode of operation of CLI module 34 of FIG. 3 to perform address resolution consistent with the principles of the invention. Upon receiving a command, or operational request, that produces an address or addresses as output (60), CLI module 34 invokes the appropriate system module 36 via management server module 32 to perform the desired operation, *e.g.*, obtaining a list of network peers (62). CLI module 34 determines which system module 36 to invoke based on the type of operation requested. CLI module 34 then receives the address or addresses in a self-describing data format, such as XML-tagged output (64). With the address or addresses expressed in a self-describing data format, CLI module 34 can readily parse the output of system module 36 and identify any numeric addresses contained in the output.

CLI module 34 then submits a query to a name server, such as a DNS server (66). This query contains the numeric address or addresses, which may be rendered in ASCII format. For example, the query may take the form of a “resolve” command sent to the DNS server. The DNS server performs a lookup operation to resolve each numeric address to a symbolic name (68) and returns a list of symbolic names to CLI module 34. CLI module 34, in turn, returns these symbolic names to the client that issued the original command (70). Accordingly, name resolution is handled by CLI module 34 and the DNS server, rather than by system module 36. As a result, system module 36 does not become blocked, and is free to process other tasks.

Not all numeric addresses are necessarily resolved to names. For example, if one system module 36 presents numeric addresses to another system module 36 rather than a human user, name resolution may not be necessary. In one embodiment, CLI module 34 determines which addresses to resolve on a case-by-case basis, rather than attempting to resolve every tagged numeric address.

Further, CLI module 34 can resolve addresses in whole or in part. As a specific example, port numbers may be left unresolved. Thus, CLI module 34 may resolve the numeric address

<address> 10.0.0.1:179 </address>

to the symbolic name

example.domain.net:179.

5 Addresses output in a self-describing format by system modules 36 may be used for purposes other than resolution to symbolic names. For example, an address can be used to look up a database to determine the owner associated with the address. Alternatively, CLI module 34 can use the address to look up a router policy database to determine which policies are in effect for a particular address. In general, the invention is applicable to any operation that involves using a remote server to obtain information about a numeric address.

10 A number of implementations and embodiments of the invention have been described. Nevertheless, it is understood that various modifications can be made without departing from the spirit and scope of the invention. For example, while some embodiments have been described using the specific example of XML tagged output, the invention is not limited to use with this particular format. Further, the principles of the invention can be applied in contexts other than address resolution. As specific examples, self-describing output can be used to highlight error messages or to selectively hide information to which a user is not privy. Accordingly, these and other embodiments are within the scope of the 15 following claims.

45 40 35 30 25 20 15 10 5 0